Accessing the Rako HUB from an external application.

Contents

1 Introduction

The Rako HUB can be accessed via a TCP socket or the legacy interfaces specified in ‘Accessing The Rako Bridge’. The API is intended for controlling Rako products from an external system. HTTP commands are available via the Rako Bridge API. The WRA-232 should be used for sending outbound 232/TCP/UDP messages from Rako.

2 TCP

Products:

RK-HUB
WK-HUB

The TCP interface is available on port 9762. It can be used in BASIC or JSON mode. After the client connects, you must use the **SUB** message to select the protocol. Line termination can be the following: ‘\r’, ‘\n’ or ‘\r\n’. The encoding is UTF-8. Note: The legacy protocol uses port 9761 which is described in ‘Accessing The Rako Bridge’.

Client to HUB commands are the following:

SUB: Must be sent as the first message to identify the client. Selects either BASIC or JSON.
STATUS: HUB identifiers and version info.
SEND: Transmit commands from the HUB.
QUERY: Query room data and levels info.

HUB to client feedback is provided in the following two ways:

Feedback: Triggered by “Button presses” may not result in a level change. Outputs all Rako commands.

Tracker: Occur when the level or scene of a dimmer changes. Only outputs level and scene changes.

3 BASIC

BASIC Commands (Client -> Hub)

For details on the JSON Connection protocol, please refer to the corresponding section later in this document.

SUB

Name	**SUB (BASIC)**
Parameters	<table><tr><th>Arg</th><th>Name</th><th>Example</th></tr><tr><td>0</td><td>Command</td><td>SUB</td></tr><tr><td>1</td><td>Format</td><td>BASIC</td></tr><tr><td>2</td><td>Version</td><td>V4</td></tr><tr><td>3</td><td>Client name</td><td>MY_DRIVER</td></tr><tr><td>4</td><td>Subscriptions</td><td>TF</td></tr></table>
Description	Subscriptions: **T**: Tracker events **F**: Feedback events
Request Example	SUB,BASIC,V4,MY_CONTROL_SYSTEM\r\n SUB,BASIC,V4,MY_CONTROL_SYSTEM,TF\r\n
Example Description	Subscribe as MY_CONTROL_SYSTEM. Basic format Version 4 to Tracker and Feedback.
Example Response	A,1

STATUS

<table>
<tr><th>Name</th><th>STATUS (BASIC)</th></tr>
<tr><td><b>Request</b></td><td>
<table>
<tr><th>Arg</th><th>Name</th><th>Example</th></tr>
<tr><td>0</td><td>Status</td><td>STATUS</td></tr>
<tr><td>1</td><td>RFU</td><td>0</td></tr>
</table>
</td></tr>
<tr><td><b>Request Example</b></td><td>STATUS,0\r\n</td></tr>
<tr><td><b>Response</b></td><td>
<table>
<tr><th>Arg</th><th>Name</th><th>Example</th></tr>
<tr><td>0</td><td>Command</td><td>STATUS</td></tr>
<tr><td>1</td><td>Product Type</td><td>Hub</td></tr>
<tr><td>2</td><td>Protocol Version</td><td>V4</td></tr>
<tr><td>3</td><td>HUB Id
36 character UUID</td><td>3ab4696c-8844-4ab9-a48c-2c95fcb5f0e3</td></tr>
<tr><td>4</td><td>Mac</td><td>70:B3:D5:08:40:00</td></tr>
<tr><td>5</td><td>HUB Version</td><td>3.6.8</td></tr>
<tr><td>6</td><td>RFU</td><td>0</td></tr>
</table>
</td></tr>
<tr><td><b>Example Response</b></td><td>STATUS,Hub,V4,353e89dd-862c-3256-aa79-8c3b89c23f21,70:B3:D5:08:40:00,3.6.8,0\r\n</td></tr>
</table>

SEND

<table>
<tr><th>Name</th><th>SEND (BASIC)</th></tr>
<tr><td>Request</td><td>
<table>
<tr><th>Arg</th><th>Name</th><th>Example</th></tr>
<tr><td>0</td><td>Command</td><td>SEND</td></tr>
<tr><td>1</td><td>Room</td><td>4</td></tr>
<tr><td>2</td><td>Channel</td><td>1</td></tr>
<tr><td>3</td><td>Command</td><td>SCENE (See commands table)</td></tr>
<tr><td>4..7</td><td>Argument(s)</td><td>4</td></tr>
</table>
</td></tr>
<tr><td>Request Example</td><td>SEND,4,1,SCENE,5\r\n
SEND,1,2,LEVEL,3\r\n
SEND,5,4,RGB,127,0,255\r\n</td></tr>
<tr><td>Response</td><td>
<table>
<tr><th>Arg</th><th>Name</th><th>Example</th></tr>
<tr><td>0</td><td>Command</td><td>SEND</td></tr>
<tr><td>1</td><td>Success</td><td>1</td></tr>
</table>
</td></tr>
<tr><td>Example Response</td><td>SEND,1
AERROR,UNKNOWN_ERROR</td></tr>
<tr><td>Response Notes</td><td>The result ‘1’ indicates the command has been successfully queued. It may fail to execute when processed. It is recommended to check the tracker feedback to ensure the command is performed.</td></tr>
</table>

BASIC Commands

<table>
<tr><th>Command</th><th>Description</th></tr>
<tr><td>SCENE</td><td>Scene 0-16 (0 = off)</td></tr>
<tr><td>LEVEL</td><td>Level (0-255)</td></tr>
<tr><td>FADE_UP</td><td>Start fading up</td></tr>
<tr><td>FADE_DOWN</td><td>Start fading down</td></tr>
<tr><td>FADE_STOP</td><td>Stop fading</td></tr>
<tr><td>STORE</td><td>Store currently selected scene / levels</td></tr>
<tr><td>RGB</td><td>
<table>
<tr><th>Arg</th><th>Name</th><th>Example</th></tr>
<tr><td>4</td><td>Red</td><td>255</td></tr>
<tr><td>5</td><td>Green</td><td>0</td></tr>
<tr><td>6</td><td>Blue</td><td>255</td></tr>
<tr><td>7</td><td>[Optional] ColorSendType</td><td>0</td></tr>
</table>
<p>White values are automatically calculated from the RGB.</p>
<p>Colour send type:</p>
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0</td><td>SEND_COLOR_AND_LEVEL</td></tr>
<tr><td>1</td><td>SEND_COLOR_ONLY</td></tr>
</table>
</td></tr>
<tr><td>KELVIN</td><td>
<table>
<tr><th>Arg</th><th>Name</th><th>Example</th></tr>
<tr><td>4</td><td>Kelvin</td><td>2700</td></tr>
</table>
</td></tr>
</table>

RGB example

RGB send/Example Response with RGBW strip on room 85 channel 6.
Connection subscribed (SUB) with ‘TF’ flags.

```
SEND,85,6,RGB,150,255,255
F,85,8,LEVEL,123
F,85,6,LEVEL,210
F,85,7,LEVEL,210
F,85,9,LEVEL,255
SEND,1
T,85,8,LEV,150,123,168,0
T,85,6,LEV,255,210,280,0
T,85,7,LEV,0,210,1310,0
T,85,9,LEV,0,255,1591,0
```

User sends the ‘SEND’ command.
Hub replies with 4 ‘F’ feedback commands are sent with roughly 500ms delay.
Hub replies with SEND,1.
Hub replies with 4 ‘T’ tracker messages showing current and target levels.

QUERY

Strings with invalid characters for CSV will be escaped, when this happens the entire string will be quoted.

<table>
<tr><th>Name</th><th>QUERY (BASIC)</th></tr>
<tr><td><b>Request</b></td><td>
<table>
<tr><th>Arg</th><th>Name</th><th>Example</th></tr>
<tr><td>0</td><td>Command</td><td>QUERY</td></tr>
<tr><td>1</td><td>Type</td><td>ROOM</td></tr>
<tr><td>2</td><td>Room Id (Optional. If missing uses room 0 (All rooms))</td><td>17</td></tr>
<tr><td>3</td><td>Arguments (Optional)</td><td>N</td></tr>
</table>
Arguments
<b>N: No header</b>
</td></tr>
<tr><td><b>Response</b></td><td>
<table>
<tr><th>Line</th><th>Name</th><th>Example</th></tr>
<tr><td>0</td><td>Column headers</td><td>QUERY,ROOM_ID,ROOM_TITLE,ROOM_TYPE\r\n</td></tr>
<tr><td>1..x</td><td>Data</td><td>17,Master Bedroom,LIGHT\r\n</td></tr>
</table>
</td></tr>
<tr><td><b>Note</b></td><td>Type can be ROOM, CHANNEL or LEVEL. See tables below.</td></tr>
</table>

Room Query

<table>
<tr><td><b>Request Example</b></td><td>QUERY,ROOM\r\n
QUERY,ROOM,0,N\r\n
QUERY,CHANNEL,0\r\n
QUERY,LEVEL,17,N\r\n
QUERY,RGB_LEVEL,17</td></tr>
<tr><td><b>Example Response (ROOM)</b></td><td>QUERY_HEADER,ROOM_ID,ROOM_TITLE,ROOM_TYPE,ROOM_MODE\r\n
QUERY,17,Master Bedroom,LIGHT,0\r\n
QUERY,14,Master Dress,BLIND,0\r\n
QUERY,13,"One Room",LIGHT,0\r\n
QUERY_COMPLETE,3</td></tr>
<tr><td><b>Response description (ROOM)</b></td><td>
<table>
<tr><th>Argument</th><th>Name</th><th>Example</th></tr>
<tr><td>0</td><td>ROOM_ID</td><td>17</td></tr>
<tr><td>1</td><td>ROOM_TITLE</td><td>Master Bedroom</td></tr>
<tr><td>2</td><td>ROOM_TYPE</td><td>LIGHT</td></tr>
<tr><td>3</td><td>ROOM_MODE
<i>(Added in V3)</i></td><td>4+OFF</td></tr>
</table>
</td></tr>
</table>

ROOM_TYPE

<table>
<tr><td><b>LIGHT</b></td><td>Scene 1-4, Fade Up, Fade Down (Fade buttons should send stop on release)</td></tr>
<tr><td><b>BLIND</b></td><td>3 buttons open,stop,close.
<table>
<tr><th>Button</th><th>Press</th><th>Release</th><th>Alt</th></tr>
<tr><td>Open</td><td>FadeUp</td><td>Stop</td><td>Scene 2</td></tr>
<tr><td>Stop</td><td></td><td>Scene 3</td><td></td></tr>
<tr><td>Close</td><td>Fade Down</td><td>Stop</td><td>Scene 4</td></tr>
</table>
If press/release events are unavailable use Alt.</td></tr>
<tr><td><b>SWITCH</b></td><td>On: Scene 1
Off: Scene 0 (Off)</td></tr>
<tr><td><b>CURTAIN</b></td><td>Same as Blind (Different room icon)</td></tr>
<tr><td><b>AUDIO</b></td><td>Deprecate RFU</td></tr>
<tr><td><b>BLIND_SMART</b></td><td>Positional blind with level percent used for blind position</td></tr>
<tr><td><b>CURTAIN_SMART</b></td><td>Same as BLIND SMART (Different room icon)</td></tr>
<tr><td><b>VENTILATION</b></td><td>Same as LIGHT (Different room icon)</td></tr>
</table>

ROOM_MODE

0	Default (None set)
S4OFF	4 Scenes, fade up/down, off
S7OFF	7 Scenes, fade up/down, off
S16OFF	16 Scenes, fade up/down, off
SNAMEDOFF	Named Scenes, fade up/down, off. Display only scenes outputted from query,scenes

Channel Query

<table>
<tr><td><b>Request Example</b></td><td>QUERY,CHANNEL\r\n
QUERY,CHANNEL,17\r\n</td></tr>
<tr><td><b>Example Response (CHANNEL)</b></td><td>QUERY_HEADER,ROOM_ID,ROOM_TITLE,ROOM_TYPE,ROOM_MODE,CHANNEL_ID,CHANNEL_TITLE,CHANNEL_TYPE,S1,S2,S3,S4,S5,S6,S7,S8,S9,S10,S11,S12,S13,S14,S15,S16\r\n
QUERY,17,Master Bedroom,Lights,1,Ceiling,SLIDER,79,191,127,63,0,0,0,0,0,0,0,0,0,0,0,0\r\n
QUERY,17,Master Bedroom,Lights,2,Bedsides,SLIDER,79,191,127,63,0,0,0,0,0,0,0,0,0,0,0,0\r\n
QUERY_COMPLETE,2\r\n</td></tr>
<tr><td><b>Response description (CHANNEL)</b></td><td>
<table>
<tr><th>Argument</th><th>Name</th><th>Example</th></tr>
<tr><td>0</td><td>ROOM_ID</td><td>17</td></tr>
<tr><td>1</td><td>ROOM_TITLE</td><td>Master Bedroom</td></tr>
<tr><td>2</td><td>ROOM_TYPE
<i>(Changed in V3)</i></td><td>LIGHT</td></tr>
<tr><td>3</td><td>ROOM_MODE
<i>(Added in V3)</i></td><td>4+OFF</td></tr>
<tr><td>4</td><td>CHANNEL_ID</td><td>5</td></tr>
<tr><td>5</td><td>CHANNEL_TITLE</td><td>Pendant</td></tr>
<tr><td>6</td><td>CHANNEL_TYPE</td><td>BLIND</td></tr>
<tr><td>7..23</td><td>S1..S16</td><td>255</td></tr>
</table>
<b>Channel Types:</b>
SLIDER
BLIND
SWITCH
SCENES
FADEUPDOWN

<b>S1..S16</b>
Scene 1-16 for each channel (0-255)
</td></tr>
<tr><td><b>Note</b></td><td>Query channel contains all data in QUERY,ROOM</td></tr>
</table>

CHANNEL_TYPE

<table>
<tr><td><b>DEFAULT</b></td><td>The type is based on the RoomType:<pre>
switch(roomType){
      case BLIND:
      case CURTAIN:
            return BLIND;
      case SWITCH:
            return SWITCH;
      default:
            return SLIDER;
}
</pre></td></tr>
<tr><td><b>SLIDER</b></td><td>Slider displaying 0 - 100% with data 0-255</td></tr>
<tr><td><b>BLIND</b></td><td>3 buttons open,stop,close.<table>
<tr><th>Button</th><th>Press</th></tr>
<tr><td>Open</td><td>FadeUp</td></tr>
<tr><td>Stop</td><td>Scene 0 or 3</td></tr>
<tr><td>Close</td><td>Fade Down</td></tr>
</table>The Rako iOS and Android app by default send scene 0 (Off) but have an option to send scene 3. Ideally stop on release for Open/Close but if not available can be missed.</td></tr>
<tr><td><b>SWITCH</b></td><td>On: Scene 1
Off: Scene 0 (Off)</td></tr>
<tr><td colspan="2"><b>Deprecated Types</b></td></tr>
<tr><td><b>SCENES</b></td><td>Off &amp; Scenes 1-4.</td></tr>
<tr><td><b>FADEUPDOWN</b></td><td>Fade Up, Fade Down. Stop on release (required).</td></tr>
</table>

Level Query

<table>
<tr><td><b>Example Response (LEVEL)</b></td><td><code>QUERY,LEVEL,18\r\n</code>
<code>QUERY_HEADER,ROOM_ID,CHANNEL_ID,CURRENT_SCENE,CURRENT_LEVEL,TARGET_LEVEL,RFU\r\n</code>
<code>QUERY,18,0,-1,0,0,0\r\n</code>
<code>QUERY,18,2,-1,50,50,0\r\n</code>
<code>QUERY_COMPLETE,2\r\n</code></td></tr>
<tr><td><b>Response description (LEVEL)</b></td><td>
<table>
<tr><th>Arg</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>ROOM_ID</td><td>Room number</td></tr>
<tr><td>1</td><td>CHANNEL_ID</td><td>Channel number</td></tr>
<tr><td>2</td><td>CURRENT_SCENE</td><td>Current scene. Can be -1 if not set.</td></tr>
<tr><td>3</td><td>CURRENT_LEVEL</td><td>Current level</td></tr>
<tr><td>4</td><td>TARGET_LEVEL</td><td>Target Level</td></tr>
<tr><td>5</td><td>RFU</td><td>Refrain from use</td></tr>
</table>
</td></tr>
</table>

Scene Query

<table>
<tr><td><b>Example Response (SCENE)</b></td><td><code>QUERY,SCENE,2\r\n</code>
<code>QUERY_HEADER,ROOM_ID,SCENE_ID,SCENE_TITLE\r\n</code>
<code>QUERY,2,1,Casual\r\n</code>
<code>QUERY,2,2,Formal\r\n</code>
<code>QUERY,2,3,Calming\r\n</code>
<code>QUERY,2,4,Candles\r\n</code>
<code>QUERY_COMPLETE,4r\n</code></td></tr>
<tr><td><b>Response description (SCENE)</b></td><td><table>
<tr><th>Arg</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>ROOM_ID</td><td>Room number</td></tr>
<tr><td>1</td><td>SCENE_ID</td><td>Scene number</td></tr>
<tr><td>2</td><td>SCENE_TITLE</td><td>Scene title</td></tr>
</table></td></tr>
</table>

Colour Query

Example Response (RGB)

```
QUERY_HEADER,ROOM_ID,ROOM_TITLE,CHANNEL_ID,CHANNEL_TITLE,TYPE\r\n
QUERY,5,Living Room,7,Wall,RGB\r\n
QUERY,5,Living Room Room,10,Slats,RGB\r\n
QUERY,5,DebugRoom,1,ABC,RGB\r\n
QUERY,5,Living Room,3,Channel3,COLORTEMPERATURE\r\n
QUERY_COMPLETE,4\r\n
```

Response description (RGB)

Arg	Name	Description
0	ROOM_ID	Room number
1	ROOM_TITLE	Room title
2	CHANNEL_ID	Channel number
3	CHANNEL_TITLE	Channel title
4	TYPE	**Current V4** RGB, COLORTEMPERATURE **Deprecated in V3** RED, GREEN, BLUE or WHITE

Note:

- The API has changed to only use RGB and COLORTEMPERATURE. Specific colours should be ignored. They are still included for backwards compatibility.
- RGB areas are shown as colour wheels in the Rako app
- Items will contain RGB or RGBW
- Internally colour wheels are indexed by the room & channel value of RED.
- Send commands to hub using the SEND,RGB command (White channel is automatically calculated)

Colour Level Query

<table>
<tr><th>Name</th><th>Color Level Query</th></tr>
<tr><td><b>Description</b></td><td>Lists all colour capable channels</td></tr>
<tr><td><b>Request Example</b></td><td><code>QUERY,COLOR_LEVEL\r\n</code></td></tr>
<tr><td><b>Feedback Example</b></td><td><code>QUERY_HEADER,ROOM_ID,CHANNEL_ID,TYPE,LEVEL,VAL1,VAL2,VAL3\r\n</code>
<code>QUERY,5,7,RGB,255,85,8\r\n</code>
<code>QUERY,27,1,RGB,0,0,0\r\n</code>
<code>QUERY,5,10,RGB,255,82,5\r\n</code>
<code>QUERY_COMPLETE,3\r\n</code></td></tr>
<tr><td><b>Data</b></td><td>
<table>
<tr><th>Arg</th><th>Name</th><th>Example</th></tr>
<tr><td>0</td><td>ROOM_ID</td><td>Room number</td></tr>
<tr><td>1</td><td>CHANNEL_ID</td><td>Channel number</td></tr>
<tr><td>2</td><td>TYPE</td><td>RGB or COLORTEMPERATURE</td></tr>
<tr><td>3</td><td>LEVEL</td><td></td></tr>
<tr><td>4</td><td>VAL1</td><td>Red or Kelvin</td></tr>
<tr><td>5</td><td>VAL2</td><td>Green</td></tr>
<tr><td>6</td><td>VAL3</td><td>Blue</td></tr>
</table>
</td></tr>
</table>

BASIC Feedback (Hub -> Client)

Feedback

<table>
<tr><th>Name</th><th>Feedback</th></tr>
<tr><td><b>Feedback Example</b></td><td><code>F,17,0,SCENE,1</code></td></tr>
<tr><td><b>Data</b></td><td>
<table>
<tr><th>Arg</th><th>Name</th><th>Example</th></tr>
<tr><td>0</td><td>Feedback</td><td>F</td></tr>
<tr><td>1</td><td>Room number</td><td>17</td></tr>
<tr><td>2</td><td>Channel number</td><td>0</td></tr>
<tr><td>3</td><td>Type</td><td>SCENE</td></tr>
<tr><td>4</td><td>Argument</td><td>1</td></tr>
</table>
</td></tr>
</table>

Tracker

Name	**Tracker (LEV)**
Tracker Example	`T,17,8,LEV,127,79,299,0`
Data	

Arg	Name	Example
0	Tracker	T
1	Room number	17
2	Channel number	8
3	Type	LEV
4	Current Level	127
5	Target Level	79
6	Time to take (ms)	299
7	Temporary	0

Note:

- Temporary indicates it has come from a source such as fade buttons and is likely to not progress to the target level. If unsure, ignore events with temporary=1.

Name	**Tracker (RGB)**
Tracker Example	`T,17,8,RGB,127,79,250`
Data	

Arg	Name	Example
0	Tracker	T
1	Room number	17
2	Channel number	8
3	Type	RGB
4	Red	127
5	Green	79
6	Blue	250

Note:

- Red, Green, Blue values are 0-255
- The channel number represents the red channel in the RGB group

Name	**Tracker (COLORTEMPERATURE)**
Tracker Example	`T,17,8,COLORTEMPERATURE,6500`
Data	

Arg	Name	Example
0	Tracker	T
1	Room number	17
2	Channel number	8
3	Type	COLORTEMPERATURE
4	Kelvin	127

BASIC Notes

- The column names will only change with a new version specified with the ‘SUB’ command. It is not required to check the ‘Column headers’.

4 JSON

Connection (JSON)

Name	**SUB (JSON)**
Direction	Client->Hub
Parameters	<table><tr><th>Arg</th><th>Name</th><th>Example</th></tr><tr><td>0</td><td>Command</td><td>SUB</td></tr><tr><td>1</td><td>Format</td><td>JSON</td></tr><tr><td>2</td><td>JSON Structure</td><td></td></tr></table>
Description	Subscriptions: **TRACKER**: Tracker events **FEEDBACK**: Feedback events
Example	`SUB,JSON,{"version": 2, "client_name": "MY_TEST_CLIENT", "subscriptions": ["TRACKER", "FEEDBACK"] }\r\n`
Example Description	Subscribe as MY_TEST_CLIENT JSON format version 2 to Tracker and Feedback.

STATUS (JSON)

<table>
<tr><th>Name</th><th>STATUS (JSON)</th></tr>
<tr><td><b>Direction</b></td><td>Client-&gt;Hub</td></tr>
<tr><td><b>Description</b></td><td>Subscriptions:
<b>TRACKER</b>: Tracker events
<b>FEEDBACK</b>: Feedback events</td></tr>
<tr><td><b>Example</b></td><td><code>{   "name": "status",</code>
<code>   "payload": {}</code>
<code>}</code></td></tr>
<tr><td><b>Response</b></td><td>
<table>
<tr><th>Name</th><th>Example</th></tr>
<tr><td>productType</td><td>Hub</td></tr>
<tr><td>protocolVersion</td><td>2</td></tr>
<tr><td>hubId
36 character UUID</td><td>ebbe7961-7abb-3aed-9fef-0bb7871ef74d</td></tr>
<tr><td>mac</td><td>70:B3:D5:08:40:00</td></tr>
<tr><td>hubVersion</td><td>3.0.6</td></tr>
</table>
</td></tr>
<tr><td><b>Example Response</b></td><td><code>{"name":"status","payload":{"productType":"Hub","protocolVersion":2,"hubId":"ebbe7961-7abb-3aed-9fef-0bb7871ef74d","mac;":"70:B3:D5:08:43:27","hubVersion":"3.1.5"}}</code></td></tr>
</table>

QUERY (JSON)

<table>
<tr><td><b>Name</b></td><td><b>QUERY (JSON)</b></td></tr>
<tr><td><b>Direction</b></td><td>Client-&gt;Hub</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>Arg</td><td>Name</td><td>Example</td></tr>
<tr><td>0</td><td>queryType</td><td>ROOM</td></tr>
<tr><td>1</td><td>roomId</td><td>15</td></tr>
</table>
</td></tr>
<tr><td><b>Description</b></td><td>Queries:
<b>ROOM</b>: Query room names and types
<b>SCENE:</b> Query scene names
<b>CHANNEL</b>: Query room data and channel data
<b>SCENECHANNEL:</b> Query SCENE and CHANNEL
<b>LEVEL</b>: Query ‘live’ tracked level information
<b>COLOR</b>: Query colour capabilities of the channel</td></tr>
</table>

Query Requests

<table>
<tr><td><b>Name</b></td><td><b>QUERY: ROOM</b></td></tr>
<tr><td><b>Description</b></td><td>List rooms without channels and scenes</td></tr>
<tr><td><b>Example Request</b></td><td><pre>
{   "name": "query",
       "payload": {
             "queryType": "ROOM",
             "roomId": 0
       }
}
</pre></td></tr>
<tr><td><b>Response</b></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>roomId</td><td>Number</td></tr>
<tr><td>title</td><td>String</td></tr>
<tr><td>type</td><td>ROOM_TYPE</td></tr>
</table>
</td></tr>
<tr><td><b>Example Description</b></td><td>Query roomId 0 will return all rooms.</td></tr>
<tr><td><b>Example Response</b></td><td><pre>
{
  "name": "query_ROOM",
</pre></td></tr>
</table>

```
  "payload": [
    {
      "roomId": 9,
      "title": "Test room",
      "type": "LIGHT"
    },
    {
      "roomId": 10,
      "title": "Test room",
      "type": "CURTAIN"
    }
  ]
}
```

Name	QUERY: CHANNEL
Description	Lists rooms with channels

Example Request	`{      "name": "query",`
	`        "payload": {`
	`              "queryType": "CHANNEL",`
	`              "roomId": 0`
	`        }`
	`}`

Response	

Name	Description
channelId	Number
title	String
type	CHANNEL_TYPE
sceneLevels	Array of Scene 0-16 levels. The level will be between 0-255. This field is Deprecated and will be set to null in the future
colorType	The colour capability of the channel. This will only be set for the first channel in a multichannel sequence. Values: ● `COLORTEMPERATURE` ● `RGB` ● `null` It is recommended to use this field to determine if the Channel has colour capabilities.

colorTitle	**Single Channel Mode:** Channel Title **Multi Channel Mode:** Extracted title or Channel X This field is always provided with color title.
multiChannelComponent	This channel is part of a sequence of multiple channels that make up a RGB channel. If this value is set then level command will control the individual component colour. If this value is null, level commands will control the whole colour channel.

MultiChannelComponentColor:

- RED
- GREEN
- BLUE
- WHITE
- WHITE_WARM
- WHITE_COOL

Example Response

```
{
  "name": "query_CHANNEL",
  "payload": [
    {
      "roomId": 40,
      "title": "E+F",
      "type": "LIGHT",
      "mode": "SNAMEDOFF",
      "channel": [
        {
          "channelId": 1,
          "title": "Main Kitchen",
          "type": "SLIDER",
          "sceneLevels": null,
          "colorType": "RGB",
          "colorTitle": "Main Kitchen",
          "multiChannelComponent": null
        },
        {
          "channelId": 4,
          "title": "Kitchen island",
          "type": "SLIDER",
          "sceneLevels": null,
          "colorType": "COLORTEMPERATURE",
          "colorTitle": "Kitchen island",
          "multiChannelComponent": null
        },
        {
          "channelId": 8,
          "title": "Red Wall",
          "type": "SLIDER",
          "sceneLevels": null,
          "colorType": "RGB",
          "colorTitle": "Wall",
          "multiChannelComponent": "RED"
        },
        {
          "channelId": 9,
          "title": "Green Wall",
          "type": "SLIDER",
          "sceneLevels": null,
          "colorType": null,
          "colorTitle": "Wall",
          "multiChannelComponent": "GREEN"
        },
        {
          "channelId": 10,
          "title": "Blue Wall",
          "type": "SLIDER",
          "sceneLevels": null ,
          "colorType": null,
          "colorTitle": "Wall",
          "multiChannelComponent": "BLUE"
        },
        {
          "channelId": 11,
          "title": "White Wall",
          "type": "SLIDER",
          "sceneLevels": null,
          "colorType": null,
          "colorTitle": "Wall",
          "multiChannelComponent": "WHITE"
        }
      ]
    }
  ]
}
```

Name	**QUERY: SCENE**
Description	Lists rooms with scenes. The scene title is returned. Scene 0 should be appended for the Off command.

Example	`{      "name": "query",`
	`       "payload": {`

<table>
<tr><td></td><td><pre>
             "queryType": "SCENE",
             "roomId": 0
       }
}
</pre></td></tr>
<tr><td><b>Response</b></td><td>

Name	Description
sceneId	Number
title	String

</td></tr>
<tr><td><b>Example Response</b></td><td><pre>
{
  "name": "query_SCENE",
  "payload": [
    {
      "roomId": 179,
      "title": "Garden",
      "type": "LIGHT",
      "mode": "S4OFF",
      "scene": [
        {
          "sceneId": 1,
          "title": "All On"
        },
        {
          "sceneId": 2,
          "title": "Wall Lights"
        },
        {
          "sceneId": 3,
          "title": "Garden"
        }
      ]
    }
  ]
}
</pre></td></tr>
</table>

<table>
<tr><td><b>Name</b></td><td><b>QUERY: SCENECHANNEL</b></td></tr>
<tr><td><b>Description</b></td><td>This includes all information in Query:Room, Query:Channel and Query:Scene.
See each table for field descriptions
<b>Note: This is our recommended query for most control systems.</b></td></tr>
<tr><td><b>Example</b></td><td><pre>
{   "name": "query",
       "payload": {
</pre></td></tr>
</table>

<table>
<tr><td></td><td>"queryType": "SCENECHANNEL",
"roomId": 0
}
}</td></tr>
<tr><td><b>Response</b></td><td>Room response with both Channel and Scene provided.
See Query Room, Channel and Scene</td></tr>
<tr><td><b>Response Example</b></td><td>{
"name": "query_SCENECHANNELL",
"payload": [
{
"roomId": 40,
"title": "E+F",
"type": "LIGHT",
"mode": "SNAMEDOFF",
"channel": [...],//SEE CHANNEL
"scene": [...]//SEE SCENE
}
]
}</td></tr>
</table>

<table>
<tr><td><b>Name</b></td><td><b>QUERY: LEVEL</b></td></tr>
<tr><td><b>Description</b></td><td>Query the Tracker information. This contains the Live state of the channels.</td></tr>
<tr><td><b>Example</b></td><td>{ "name": "query",
"payload": {
"queryType": "LEVEL",
"roomId": 0
}
}</td></tr>
<tr><td><b>Response</b></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>channelId</td><td>Number</td></tr>
<tr><td>currentLevel</td><td>Number</td></tr>
<tr><td>targetLevel</td><td>Number or null</td></tr>
<tr><td>sceneInfo</td><td>Object or null</td></tr>
</table>
<b>SceneInfo</b>
<table>
<tr><th>Name</th><th>Description</th></tr>
</table>
</td></tr>
</table>

<table>
<tr>
<td></td>
<td>
<table>
<tr><td><code>kelvin</code></td><td>Number or null</td></tr>
<tr><td><code>red</code></td><td>Number or null</td></tr>
<tr><td><code>green</code></td><td>Number or null</td></tr>
<tr><td><code>blue</code></td><td>Number or null</td></tr>
</table>
<p>Fetching the ‘live’ level should use <code>targetLevel</code> if set else use the <code>currentLevel</code>.</p>
<p>JavaScript</p>
<pre>
int liveLevel = (channel.targetLevel
!= null) ? channel.targetLevel :
channel.currentLevel;
</pre>
</td>
</tr>
<tr>
<td><b>Example Response</b></td>
<td>
<pre>
{
  "name": "query_LEVEL",
  "payload": [
    {
      "channel": [
        {
          "channelId": 0,
          "currentLevel": 50,
          "targetLevel": null,
          "sceneInfo": null
        },
        {
          "channelId": 1,
          "currentLevel": 50,
          "targetLevel": null,
          "sceneInfo": {"kelvin": 2700},
        }
      ],
      "roomId": 45,
      "currentScene": -1
    }
  ]
}
</pre>
</td>
</tr>
</table>

Name	**QUERY: COLOR**
Description	**@Deprecated** Included for backwards compatibility
Response	

Name	Description
roomId	Number
title	Number
type	Number or null
primaryChannel	[See below]
rgbMode	SINGLECHANNEL or MULTICHANNEL
rgbTitle	String
channel	Array

Channel

Name	Description
channelId	Number
title	String
rgbColor	String. The values use MultiChannelComponentColor

Example

```
{      "name": "query",
       "payload": {
             "queryType": "COLOR",
             "roomId": 0
       }
}
```

Example Description

Query roomId 0 will return all rooms.

Example Response

```
{
      "roomId": 97,
      "title": "First Room",
      "type": "LED",
      "channel": [
        {
          "channelId": 4,
          "title": "Red",
          "rgbColor": "RED"
        },
        {
          "channelId": 5,
          "title": "Green",
          "rgbColor": "GREEN"
        },
        {
          "channelId": 6,
          "title": "Blue",
          "rgbColor": "BLUE"
        }
      ],
      "primaryChannel": 4,
      "rgbMode": "MULTICHANNEL",
      "rgbTitle": ""
    },
    {
      "roomId": 55,
      "title": "Second Room",
      "type": "COLORTEMPERATURE",
      "channel": [
        {
          "channelId": 1,
          "title": "CAT",
          "rgbColor": "WHITE_WARM"
        }
      ],
      "primaryChannel": 1,
      "rgbMode": "SINGLECHANNEL",
      "rgbTitle": "CAT"
    },
```

Query notes:
Room & Channel types are defined in the BASIC section

Query,LEVEL
Tracker information is not present at HUB power-up. The information will only be provided after the room has been controlled.

QUERY, COLOR
When using RGB channels. Rako supports two types of RGB configuration. Single channel and MultiChannel. In single channel mode, the channel's level represents the brightness of the fixture. In multi-channel mode, the channel level represents one of the Red, Green, or Blue components. The RGB information is always stored on the first channel, this is referred to as the primaryChannel. The API has been designed to handle all RGB functionality through the primaryChannel, which works in both situations. This allows you to ignore the complexity and handle all RGB functionality through the primaryChannel.
It is important to note that if a room has both, the RGB channel type will take precedence. Each RGB 'fixture' is represented by a single combination of

roomId and channelId (primaryChannel). The feedback is provided against that combination.

The API simplifies the handling of single and multi channel RGB modes. This allows for a consistent way to control colour and brightness, irrespective of the underlying channel configuration. Each colour picker can be bound to a single Room/Channel combination.

<table>
<tr><th>Name</th><th>QUERY: COLOR LEVEL</th></tr>
<tr><td><b>Description</b></td><td><b>@Deprecated</b>
Included for backwards compatibility</td></tr>
<tr><td><b>Response</b></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td><code>roomId</code></td><td>Number</td></tr>
<tr><td><code>channelId</code></td><td>Number</td></tr>
<tr><td><code>currentLevel</code></td><td>Number</td></tr>
<tr><td><code>targetLevel</code></td><td>Number or null</td></tr>
<tr><td><code>levelInfo</code></td><td>Object or null</td></tr>
</table>
<b>LevelInfo</b>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td><code>kelvin</code></td><td>Number or null</td></tr>
<tr><td><code>red</code></td><td>Number or null</td></tr>
<tr><td><code>green</code></td><td>Number or null</td></tr>
<tr><td><code>blue</code></td><td>Number or null</td></tr>
</table>
</td></tr>
<tr><td><b>Example</b></td><td><pre>{   "name": "query",
       "payload": {
              "queryType": "COLOR-LEVEL",
              "roomId": 0
       }
}</pre></td></tr>
<tr><td><b>Example Description</b></td><td>Query roomId 0 will return all rooms.</td></tr>
</table>

Example Response

```
{
  "name": "query_COLOR-LEVEL",
  "payload": [
    {
      "roomId": 5,
      "channelId": 10,
      "currentLevel": 191,
      "targetLevel": null,
      "levelInfo": {
        "kelvin": null,
        "red": 255,
        "green": 255,
        "blue": 255
      },
      "targetOrCurrent": 191
    },
    {
      "roomId": 5,
      "channelId": 7,
      "currentLevel": 191,
      "targetLevel": null,
      "levelInfo": {
        "kelvin": null,
        "red": 255,
        "green": 255,
        "blue": 255
      },
      "targetOrCurrent": 191
    },
    {
      "roomId": 89,
      "channelId": 4,
      "currentLevel": 0,
      "targetLevel": 255,
      "levelInfo": {
        "kelvin": 2700,
        "red": null,
        "green": null,
        "blue": null
      },
      "targetOrCurrent": 255
    }
  ]
}
```

SEND COMMAND (JSON)

Name	**Send**
Direction	Client->Hub

Payload

Payload:

Element name	Example
`room`	Room number
`channel`	Channel number
`description`	Description
`action`	Action

Action:

Element name	Example
`command`	*required command
`scene`	***optional** scene number
`level`	***optional** level 0-255
`down`	***optional** fade direction

Example (Scene)

```
{       "name": "send",
        "payload": {
                "room": 1,
                "channel": 2,
                "action": {
                        "command": "scene",
                        "scene": 3
                }
        }
}
```

Example (Level)

```
{       "name": "send",
        "payload": {
                "room": 1,
                "channel": 2,
                "action": {
                        "command": "levelrate",
                        "level": 255
                }
        }
}
```

Example (Store)

```
{       "name": "send",
        "payload": {
                "room": 1,
                "channel": 2,
                "action": {
                        "command": "store"
                }
        }
}
```

Example (Fade Up)

```
 {
        "name": "send",
        "payload": {
                "room": 1,
                "channel": 0,
                "action": {
                        "command": "fade",
                        "down": false
                }
        }
}
```

Example (Fade Stop) *Release event for Fade Up or Fade Down*

```
 {
        "name": "send",
        "payload": {
                "room": 1,
                "channel": 0,
                "action": {
                        "command": "stop"
                }
        }
}
```

Command list (JSON)

Command	ID	Argument	Description
ident	8		Flash the circuit
store	13		Store
stop	15		Stop fading
scene	49	scene	Scene 0-16 (0 = off)
levelrate	52	level	Level (0-255)
fade	50	down	Start fading up or down.
motor-stop	80		

SEND COLOUR (JSON)

Name	**Send Colour (RGB/W)**
Direction	Client->Hub
Payload	Payload: Element name: Example `room`: Room number `channel`: Channel number `red`: Red (0-255) `green`: Green (0-255) `blue`: Blue (0-255) `colorSendType`: Colour send type. Default SEND_COLOR_AND_LEVEL `rgbExcludesBrightness`: Optional: Flag indicating if red, green and blue values exclude luminance information (brightness). `level`: Optional: Level Information (0-255). `rgbExcludesBrightness` must be set.
Example	see below

Payload:

Element name	Example
`room`	Room number
`channel`	Channel number
`red`	Red (0-255)
`green`	Green (0-255)
`blue`	Blue (0-255)
`colorSendType`	Colour send type. Default SEND_COLOR_AND_LEVEL
`rgbExcludesBrightness`	Optional: Flag indicating if red, green and blue values exclude luminance information (brightness).
`level`	Optional: Level Information (0-255). `rgbExcludesBrightness` must be set.

Example

```
{      "name": "send-color",
       "payload": {
              "room": 16,
              "channel": 2,
              "colorSendType":
"SEND_COLOR_AND_LEVEL",
              "red": 25,
              "green": 50,
              "blue": 255
       }
}
```

Name	**Send Color (Temperature)**

Payload

Payload:

Element name	Example
`room`	Room number
`channel`	Channel number
`temperature`	Kelvin value (1700-10,000)
`level`	Optional

Example

```
{      "name": "send-colorTemp",
       "payload": {
              "room": 16,
              "channel": 2,
              "colorSendType": "SEND_COLOR_ONLY",
              "temperature": 2700
       }
}
```

Example with level

```
{      "name": "send-colorTemp",
       "payload": {
              "room": 16,
              "channel": 2,
              "colorSendType":
"SEND_COLOR_AND_LEVEL",
              "temperature": 2700,
              "level": 2700,

       }
}
```

The `send-color` and `send-colorTemp` command can control the colour and intensity values independently. The API allows this to be specified in the colorSendType field.

Send RGB values with luminance information

The typical use case for sending RGB. The "red", "green", and "blue" fields generally include both colour and brightness information. The RGB values, ranging from 0 to 255, influence the overall brightness of the light.

Send RGB values without luminance information

When `rgbExcludesBrightness` is set, the “red”, “green” and “blue” fields sent represent pure colour information without any luminance (brightness) data. Internally the HUB will normalise (brightness forced to 100%) the RGB values. Using this in combination with

`SEND_COLOR_ONLY` allows control of the colour without changing the brightness.

Colour Send Type (`colorSendType`)

- `SEND_COLOR_AND_LEVEL` should be used when sending both colour and luminance (brightness) information.
- `SEND_LEVEL_ONLY` should be used when sending only brightness information, maintaining the previously set colour.
- `SEND_COLOR_ONLY` should be used when sending only colour information, maintaining the previously set brightness level.

Note:

- `SEND_COLOR_AND_LEVEL` should be used when the RGB values include luminance information.
- Only `SEND_COLOR_AND_LEVEL` is globally supported. Our product range does not yet universally support `SEND_LEVEL_ONLY` or `SEND_COLOR_ONLY`. Unsupported devices will be shown on the Log page. The supported types can be identified using the ChannelType `RGB` and `COLORTEMPERATURE`.

Feedback

Feedback (JSON)

<table>
<tr><th>Name</th><td><b>Feedback</b> (Use for mapping to button presses)</td></tr>
<tr><th>Direction</th><td>Hub->Client</td></tr>
<tr><th>Payload</th><td>
<table>
<tr><th>Element name</th><th>Example</th></tr>
<tr><td><code>room</code></td><td>Room number</td></tr>
<tr><td><code>channel</code></td><td>Channel number</td></tr>
<tr><td><code>description</code></td><td>Description</td></tr>
<tr><td><code>action</code></td><td>Action</td></tr>
</table>
Action structure
<table>
<tr><th>Element name</th><th>Example</th></tr>
<tr><td><code>command</code></td><td>*required command</td></tr>
<tr><td><code>scene</code></td><td><b>*optional</b> scene number</td></tr>
<tr><td><code>level</code></td><td><b>*optional</b> level 0-255</td></tr>
<tr><td><code>down</code></td><td><b>*optional</b> fade direction</td></tr>
</table>
</td></tr>
</table>

```
{
  "name": "feedback",
  "payload": {
    "action": {
      "actUniqueId": -1,
      "defaultFadeRate": true,
      "decay": 0,
      "level": 90,
      "expFadeRate": false,
      "command": 52
    },
    "room": 85,
    "channel": 4,
    "description": "[Rm:85 Wine Cellar][Ch:4 Counter] Level: 35%"
  }
}
```

Tracker (JSON)

<table>
<tr><td><b>Name</b></td><td><b>Tracker</b> (Use for tracking scenes and levels)</td></tr>
<tr><td><b>Direction</b></td><td>Hub->Client</td></tr>
<tr><td><b>Type</b></td><td><b>level</b>
int currentLevel
int targetLevel
int timeToTake
boolean temporary
<b>scene</b>
int scene
<b>color-rgb</b>
int[] rgb
<b>color-temperature</b>
int kelvin
int brightness</td></tr>
<tr><td><b>Payload</b></td><td>Type: level
<table>
<tr><th>Element name</th><th>Description</th></tr>
<tr><td><code>roomId</code></td><td>Room number</td></tr>
<tr><td><code>channelId</code></td><td>Channel number</td></tr>
<tr><td><code>currentLevel</code></td><td>Current Level</td></tr>
<tr><td><code>targetLevel</code></td><td>Target Level</td></tr>
<tr><td><code>timeToTake</code></td><td>Time to take in milliseconds</td></tr>
<tr><td><code>temporary</code></td><td>Temporary indicates it has come from a source such as fade buttons and is likely to not progress to the target level. If unsure, ignore events with temporary=true.</td></tr>
</table>
Type: scene
<table>
<tr><th>Element name</th><th>Description</th></tr>
<tr><td><code>roomId</code></td><td>Room number</td></tr>
<tr><td><code>channelId</code></td><td>Channel number</td></tr>
<tr><td><code>scene</code></td><td>Scene number</td></tr>
</table>
</td></tr>
</table>

**Example
(Level tracker)**
	{
	  "name": "tracker",
	  "type": "level",
	  "payload": {
	    "roomId": 85,
	    "channelId": 4,
	    "currentLevel": 127,
	    "targetLevel": 90,
	    "timeToTake": 230,
	    "temporary": false
	  }
	}

**Example 2
(Scene tracker)**
	{
	  "name": "tracker",
	  "type": "scene",
	  "payload": {
	    "roomId": 85,
	    "channelId": 0,
	    "scene": 4
	  }
	}

5 UDP/IP

Products:

RA/RTC/WA/WTC-Bridge
APR/WRE-Bridge (V1.7.9 Limited support)

UDP Discover

All text is encoded with WINDOWS-1252

To find a BRIDGE or HUB on the network send a UDP broadcast packet where the data consists of a single literal 'D'. The bridge will reply to the source IP address with:

Request

Direction	Parameters		Description	Example
	Byte	Function		
Client to UDP Broadcast	0	`0x44`: 'D' for request	Discover Required	`0x44`
Client to UDP Broadcast	1	`0x58`: 'X' for request	Bypass Disable auto detect	`0x58`
Client to UDP Broadcast	2	`0x48`: 'H' for request	Append `\r\nHub\r\n2`	`0x48`

Bridges and HUBs are discovered by the same command so it is recommended to send “DXH” as the discovery command.

Response

Line	Description	Example
0	User configurable device name	`RakoHub`
1	Mac address	`70-B3-D5-08-41-9F`
2	Product type	`HUB`
3	Minimum api version	`2`

Example

Direction	Broadcast Reply From Bridge
Bridge	`RAKOBRIDGE\r\n00-04-A3-B1-21-F1\r\n`
HUB before version 3.0.6	`RakoHub\r\n70-B3-D5-08-40-00\r\nHUB\r\n`
HUB after version 3.0.6	`MyRakoHub\r\n70-B3-D5-08-41-9F\r\nHUB\r\n2\r\n`

HUB versions after 3.0.6 include the version. The TCP API requires 3.0.6 and later.

Note
The device name is a case insensitive alphanumeric string including space and hyphen with a maximum length of 16 characters.

6 Compatibility

Hub Version	Date	Description
Pre 3.0.6		API not supported.
3.0.6	18/12/2019	JSON AND BASIC API V2 supported
3.0.9	27/04/2020	Issue present requiring extra arguments to send command (Use of version not recommended).
3.1.0	11/06/2020	BASIC API V3 supported. (V2 is still supported).
3.1.5	17/08/2020	Minimum version for JSON support
3.1.6	21/08/2020	Support for RGB feedback
3.1.7	11/12/2020	Store command added to BASIC protocol. Added scene query to BASIC and JSON.
3.6.6	14/04/2023	BASIC V4 API Colour commands require `colorSendType`
3.7.0	06/06/2023	Colour API changes
3.7.5-3.7.8 (**BETA ONLY**)	12/10/2023	JSON,QUERY,CHANNEL supportsRgb, supportsCt & secondaryMultiChannel fields added. JSON,QUERY,COLOR. Fields changed. colorTitle, colorType & multiChannelComponent. This change was temporary in the HUB beta version.
3.7.9+ (**BETA ONLY**)	14/12/2023	JSON,QUERY,COLOR updated to match V3.7.5.
3.8.2	05/03/2024	HUB firmware release (Non Beta)

Note:

- All HUBs can upgrade to the latest version.
- RA/WA/RTC/WTC/WRA Bridge will never support this protocol. Use ‘Accessing The Rako Bridge’.

7 Changelog

Version	Date	Description
0.1.5	21/01/202	API version V2
0.1.6	24/04/2020	BASIC API V3 ● Added changelog ● PLAIN renamed to BASIC (Only in document) ● Documentation for JSON API ● Added description for RoomType and ChannelType ● Added room mode to Room query ● V3 QUERY,CHANNEL Room Type was changed to be the same as QUERY,ROOM ● Added QUERY,RGB ● Added SEND,{room},{channel},RGB,{R},{G},{B}
0.1.7	20/06/2020	● Document formatting
0.1.8	10/08/2020	● Added JSON query information
0.1.9	13/08/2020	● Query example correct for Room and Channel
0.2.0	25/08/2020	● Update UDP Discovery descriptions
0.2.1	02/12/2020	● Added Store command ● Added scene query to BASIC and JSON
0.2.2	18/02/2021	● Basic tracker returns 'LEV' not 'LEVEL' ● Basic tracker 'RGB' type
0.2.3	23/04/2021	● Updated Introduction ● Added missing JSON query scene
0.2.4	31/08/2022	● Added information for JSON fade commands
0.2.6	07/02/2023	● Removed RGB_INDEX. Use Channel. ● Added send COLOR-RGB and COLOR-KELVIN
0.2.7	11/10/2023	● Deprecated JSON QUERY,COLOR-LEVEL. The HUB provides colour information with the QUERY,LEVEL command. The query type will continue to be supported. ● Deprecated JSON QUERY,COLOR. The HUB provides information via QUERY_CHANNEL. ● Changed BASIC Query,Color field title from ROOM_TYPE to CHANNEL_TYPE
0.2.8	01/12/2023	● RGB basic can now specify 'ColorSendType'
0.2.9	14/12/2023	● JSON Query,Color updated to match 3.7.0
0.3.0	12/03/2024	● HUB compatibility updated